

AD-A195 965

EOSR No. 1552

2

DTIC FILE COPY

MICROPROCESSOR CONTROL
OF
QUARTER WATT LINEAR COOLER

DTIC
ELECTE
JUN 01 1988
S D

FINAL REPORT
AUGUST 7, 1987

DECEMBER 5, 1986 TO JULY 31, 1987

Prepared under Contract Number DAAB07-87-C-F018
for Center for Night Vision and Electro-Optics

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

MAGNAVOX GOVERNMENT AND INDUSTRIAL ELECTRONICS COMPANY
ELECTRO-OPTICAL SYSTEMS, 46 INDUSTRIAL AVENUE
MAHWAH, NJ 07430

MICROPROCESSOR CONTROL
OF
QUARTER WATT LINEAR COOLER

FINAL REPORT

AUGUST 7, 1987

DECEMBER 5, 1986 TO JULY 31, 1987

Prepared under Contract Number DAAB07-87-C-F018
for Center for Night Vision and Electro-Optics

MAGNAVOX GOVERNMENT AND INDUSTRIAL ELECTRONICS COMPANY
ELECTRO-OPTICAL SYSTEMS, 46 INDUSTRIAL AVENUE
MAHWAH, NJ 07430

Approved by: 

A. Silver
Deputy Director of Engineering

TABLE OF CONTENTS

	<u>Page</u>
1.0 INTRODUCTION.....	1
1.1 MOTIVATION FOR MICROPROCESSOR/DIGITAL CONTROL.....	1
1.2 SWITCHING FREQUENCY.....	1
2.0 DESCRIPTION OF HARDWARE.....	4
2.1 COMPUTER.....	4
2.2 DIGITAL PULSE WIDTH MODULATOR (DPWM) - IXDP610.....	5
2.3 TIMING DIAGRAM.....	5
2.4 LEVEL SHIFT AND POWER AMPLIFIER.....	5
3.0 DESCRIPTION OF SOFTWARE.....	10
4.0 SURVEY OF MICROCONTROLLERS.....	13
5.0 SELECTION OF SWITCHING FREQUENCY.....	15
5.1 VOLUME CONSTRAINT.....	15
5.2 DIGITAL HARDWARE CONSTRAINT.....	15
5.3 THE SELECTION.....	15
6.0 SUMMARY.....	15
7.0 RECOMMENDATIONS FOR FURTHER STUDY.....	16

LIST OF TABLES AND ILLUSTRATIONS


<u>Figure</u>		
1	Analog Cooler Control.....	2
2	Digital/Microprocessor Cooler Control.....	3
3	Micro Controller.....	6
4	Hardware Demonstration.....	7
5	Block Diagram - IXDP610.....	8
6	Timing Diagram.....	9
7	Software Flowchart.....	11

<u>Table</u>		
1	Relative Comparison of Analog Versus Digital Design.....	4
2	Look Up Table.....	12
3	Microcontroller Survey.....	14

LIST OF APPENDICES

Appendix I - Demonstration Software

Approved For	
NES GRA81	J
DTL FIB	<input type="checkbox"/>
Other (Specify)	<input type="checkbox"/>
per ltr	



A-1

ABSTRACT

The work carried out in this study was performed by Magnavox Government & Industrial Electronics Company, Electro-Optical Systems, in accordance with the requirements of contract DAAB07-87-C-F018.^A The purpose of the program was to conduct studies of microcontroller/digital VLSI technologies for possible future use in cryogenic coolers.

A necessary condition that any electronic approach must satisfy in order to be considered for linear cooler electronics is the ability to work in the real time environment of 100 kHz or higher switching speeds. In the case of the existing analog design, this prerequisite has long been established to achieve practically sized line filters. Heretofore this speed was out of reach of microprocessor based, controllers since the state-of-the-art was somewhere in the neighborhood of 8 to 10 kHz. However, a new device on the market has provided the necessary interface hardware to bridge this gap. This advancement combined with the advances of microcontrollers in terms of cost, size, and performance suggested that an investigation of the microcontroller/digital approach for the Magnavox quarter watt cooler electronics be performed.

1.0

INTRODUCTION

The work carried out in this study was performed by Magnavox Government & Industrial Electronics Company, Electro-Optical Systems, in accordance with the requirements of contract DAAB07-87-C-F018. The purpose of the program was to conduct studies of microcontroller/digital VLSI technologies for possible future use in Magnavox cryogenic coolers.

1.1

MOTIVATION FOR MICROPROCESSOR/DIGITAL CONTROL

The present technology used for Magnavox cooler linear motor control is based on an analog circuit design. The diagram shown in Figure 1 is representative of the approach to meet a requirement for closed-loop cold station temperature control. Although this analog design functions well, there are compelling reasons to look in the direction of microcontroller/digital electronics, Figure 2, for future designs.

The first and most important reason is potential cost savings. The analog approach requires hand assembled control hybrids with expensive laser trimming. This is in contrast to the microcontroller approach that utilizes low cost, mass produced IC's.

A second potential benefit from a microcontroller approach is accuracy. A performance advantage is achieved in maintaining accurate temperature control of the detector cold station. As the tolerance on temperature is tightened the cost for accurate trimming of the analog design increases exponentially. In the case of the microcontroller, accuracies to $\pm 1^\circ\text{K}$ are easily attainable. Additionally, long term thermal drift, and component variability, are not a concern.

There are advantages in flexibility. Such things as customer requirements for the temperature set point or specialized modes of operation to conserve power can be satisfied by simple changes in software. New drive wave forms, easily generated in software, could have a favorable impact on efficiency and/or reduction of mechanical vibration.

1.2

SWITCHING FREQUENCY

With these advantages, summarized in Table 1, the natural question is why weren't microcontroller electronics used sooner? The answer is that the volume constraints on the cooler electronics requires that the switching frequency of the power amplifier be above 100 kHz. As the frequency is reduced from this level the input EMI filter components tend to grow. Since they just fit now, any growth would mean a deterioration in the performance of the filter.

The diagram illustrates a closed-loop control system for a heat exchanger. The control loop starts with a **TEMP COMMAND (VOLTS DC)** input, which is summed with a feedback signal from a **TEMP SENSE** block (gain K_4). The resulting error signal is processed by gain K_1 and then summed with a **MOTOR VOLTAGE COMMAND** input. This combined signal is summed again and passed through gain K_2 . The output of K_2 is summed with a reference signal from a **SINE WAVE GEN** (874KHZ) block, which is also fed into a **FULL WAVE RECT** block. The output of the rectifier is summed with the output of the **SINE WAVE GEN** (874KHZ) block. The resulting signal is summed with a feedback signal from a **MOTOR VOLTS (RMS)** block (gain K_5). The output of this final summing junction is processed by gain K_3 and then a **PWM** block. The PWM signal is fed into a **FULL BRIDGE AMPLIFIER**, which drives a **MOTOR**. The motor is connected to a **HEAT LOAD** and a **COLD STATION**. The **TEMP SENSE** block (gain K_4) provides feedback from the **COLD STATION** to the first summing junction. The **MOTOR VOLTS (RMS)** block (gain K_5) provides feedback from the **MOTOR** to the final summing junction.

FIGURE 1

DIGITAL/MICROPROCESSOR COOLER CONTROL

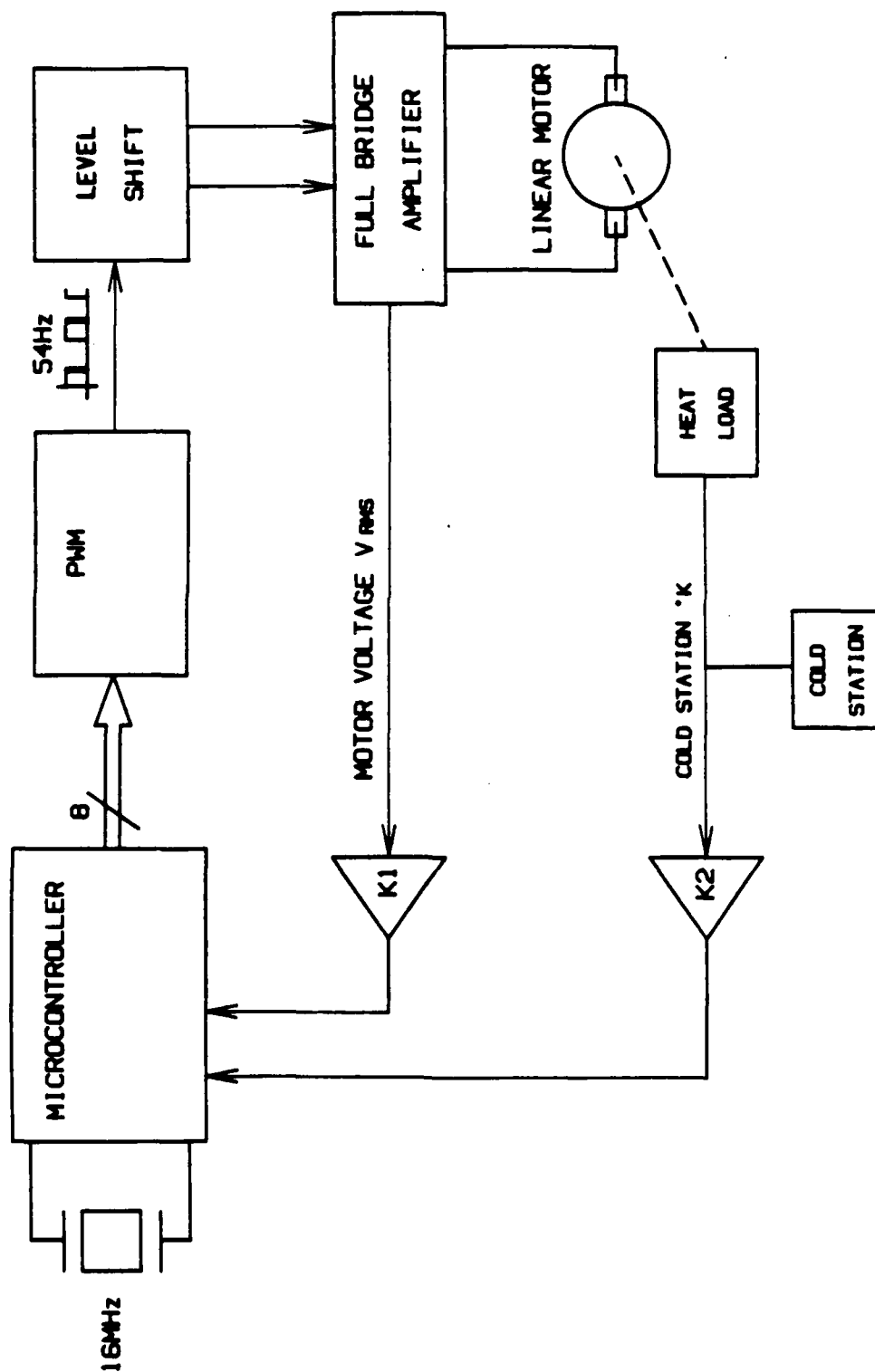


FIGURE 2

Table 1. Relative Comparison of Analog Versus Digital Design

	<u>Analog Control</u>	<u>Digital Control</u>
Parts Count	High	Low
Flexibility	Low	High
High Volume Cost	High	Low
Low Volume Cost	Low	High
Reliability	Good	Better
Accuracy	High	High

Specifically, the volume available in the cooler for the electronics is 0.7 cubic inches. The analog package as it is currently configured just fits into this volume - no margin for expansion. Of that volume approximately 20% is occupied by the input EMI filters. These components are designed for the 100 kHz switching frequency. If we were to lower the frequency by a decade, the volume occupied by the filter chokes would increase by approximately 50% to maintain the same impedance level. The filter capacitors would have to be increased by 3 to 4 times depending on ripple considerations. For these reasons, switching frequencies in the range of 10 kHz are out of the question.

As mentioned previously, until recently 100 kHz real time control was out of reach of microcontroller technology. This would require algorithmic execution speeds in the order of 10 microseconds which is out of reach for present day microcontrollers. However, a new interface PWM chip overcomes this and is a focus of attention for this study.

2.0 DESCRIPTION OF HARDWARE

2.1 COMPUTER

The design shown in Figure 2 represents a production concept for the microcontroller/digital electronics. Note that this is closed loop on both motor voltage and cold station temperature. To achieve this, the basic functions of the microcontroller are:

1. Get the appropriate PWM level from the data stored in memory.
2. Multiply that number (thereby changing the gain) to adjust for variation of motor voltage from command set point.
3. Multiply a second time (again modifying the gain) to adjust for variations in thermal loading.
4. Write to the digital PWM chip the appropriate code in accordance with the above calculation.

A typical microcontroller, as shown in Figure 3, is a single chip microcomputer. It is compatible with all system timing, internal logic, ROM, RAM and I/O. Although it is not obvious in Figure 2, the microcontroller has on board the required A/D converters for closed loop control.

For the purpose of a demonstration it was convenient to use a "single board computer" (SBC) to simulate functions of a microcontroller. The arrangement shown in Figure 4, used an Ampro SBC, with the Intel 80186 microprocessor, and was programmed by an Applied Microsystem, ES 1800 Satellite Emulator. This provided easy manipulation of the code through PL/M on the PC.

2.2 DIGITAL PULSE WIDTH MODULATOR (DPWM) - IXDP610

The ability to reach switching frequencies greater than 100 kHz is achieved by the IXDP610 Digital Pulse Width Modulator (DPWM) chip. This chip is a major breakthrough in high frequency motor control applications. Without it switching would be limited to approximately 8 kHz. As discussed previously, this would eliminate consideration of the microcontroller approach.

A block diagram of the IXDP610 (DPWM) is shown in Figure 5. The programmable, CMOS, LSI device receives digital pulse width data (8 bits) from the microcontroller and generates a TTL pulse width modulated signal. The DPWM is designed for direct control by the microcontroller. Current limiting on a cycle by cycle basis can be asserted through "Output Disable Logic" (ODIS).

2.3 TIMING DIAGRAM

The timing diagram of Figure 6 shows the digital aspects of this topology. Figure 2, shows that the master clock is crystal controlled and set at 16 MHz. This clock signal together with the appropriate input to the programmable digital PWM controller, sets the switching frequency as follows:

$$\text{PWM base period} = \text{Clock Period} \times 128$$

$$8 \text{ Microsecond (125 kHz)} = 63 \text{ nanoseconds (16 MHz)} \times 128$$

The rate at which the CPU strobes through the look up table is controlled by a programable counter which is fed by an external 125 kHz signal. The counter is set to count 25, 8 μs counts (8 μs x 25 = 200 μs). Upon decrementing to a 0 count, an interrupt is pulled. At this time the current value is taken out of the look up table, outputted to a port and the table pointer incremented. This data is outputted to the IXDP610 latch. With the "Chip Select" low, the data is written to the pulse width latch. Under the steady state mode of operation, the pulse width of the power amplifier is fixed for the next 200 μs , after which another CS pulse will allow an update.

2.4 LEVEL SHIFT AND POWER AMPLIFIER

With a TTL pulse width modulation signal from the DPWM chip, the next requirement is to amplify the signal and drive the motor. This part of the

MICRO CONTROLLER

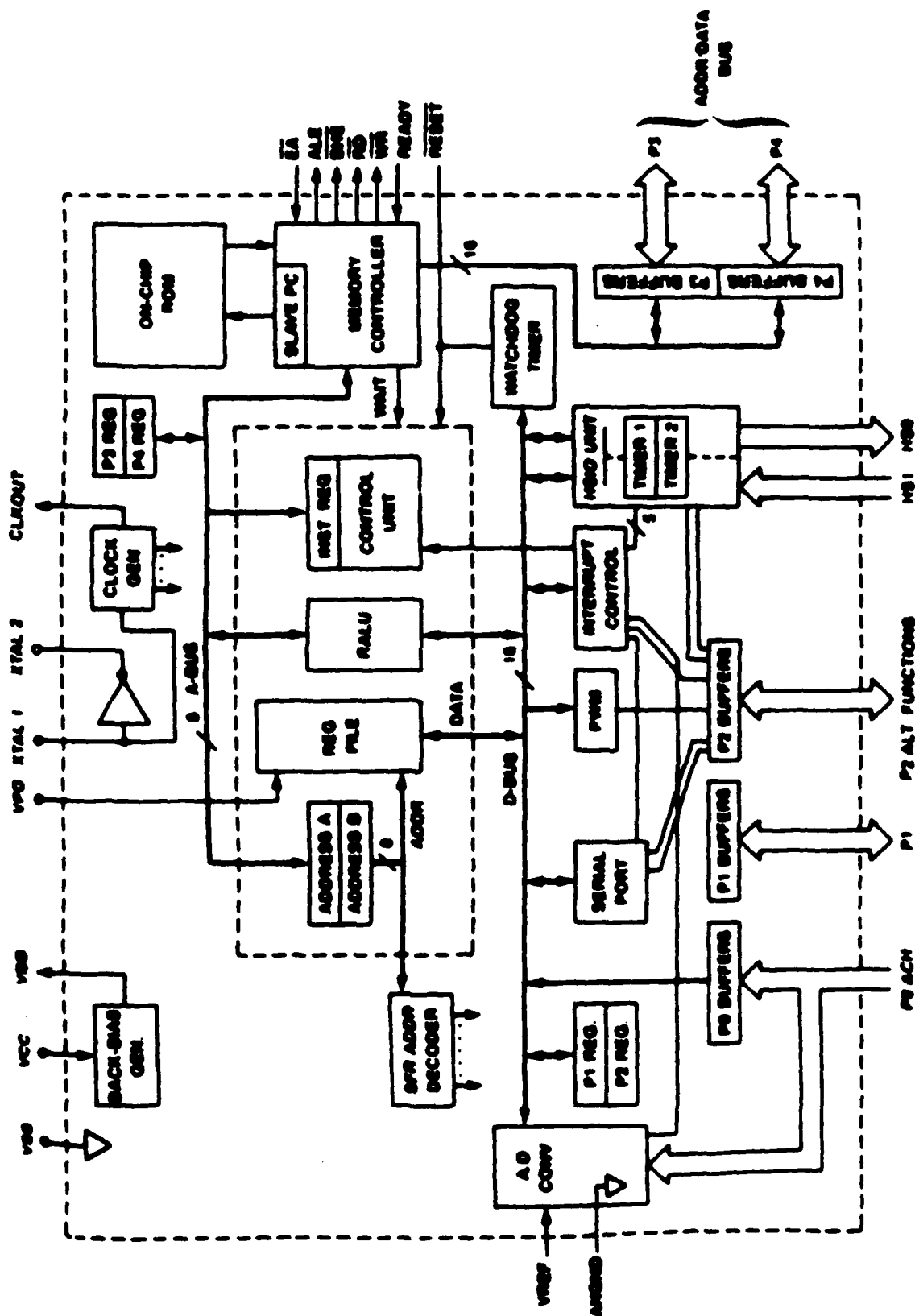


FIGURE 3

HARDWARE DEMONSTRATION

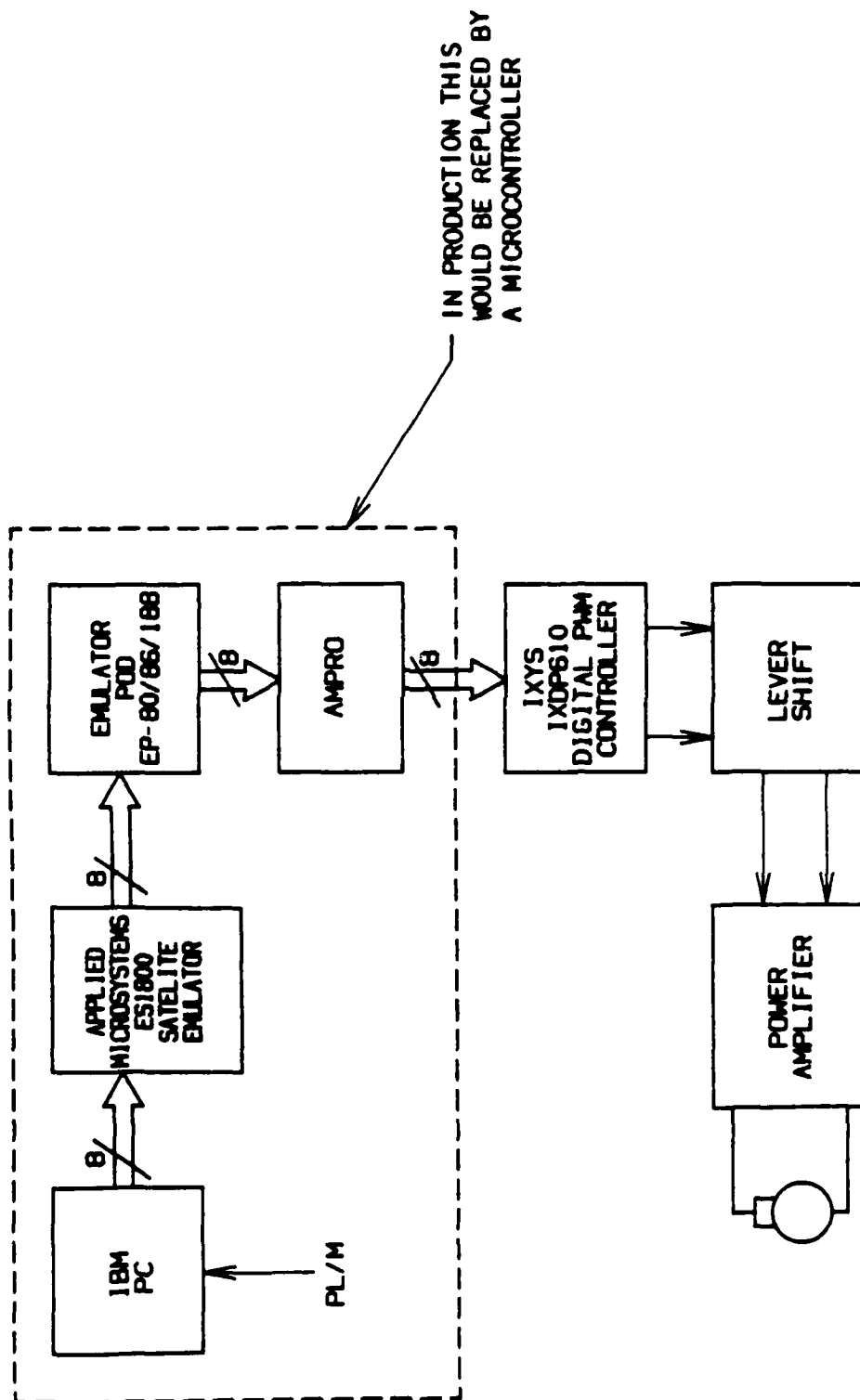


FIGURE 4

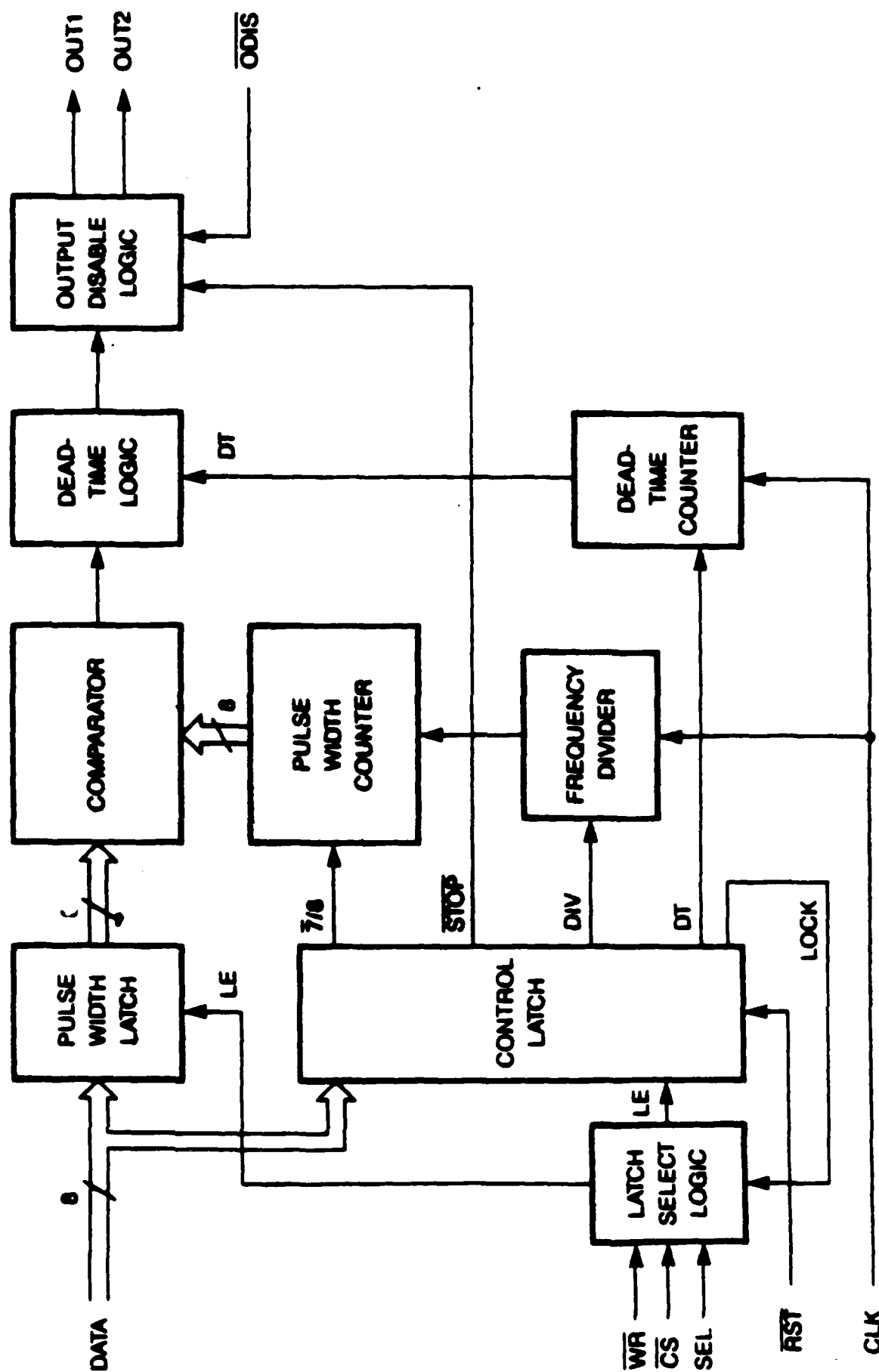


FIGURE 5.

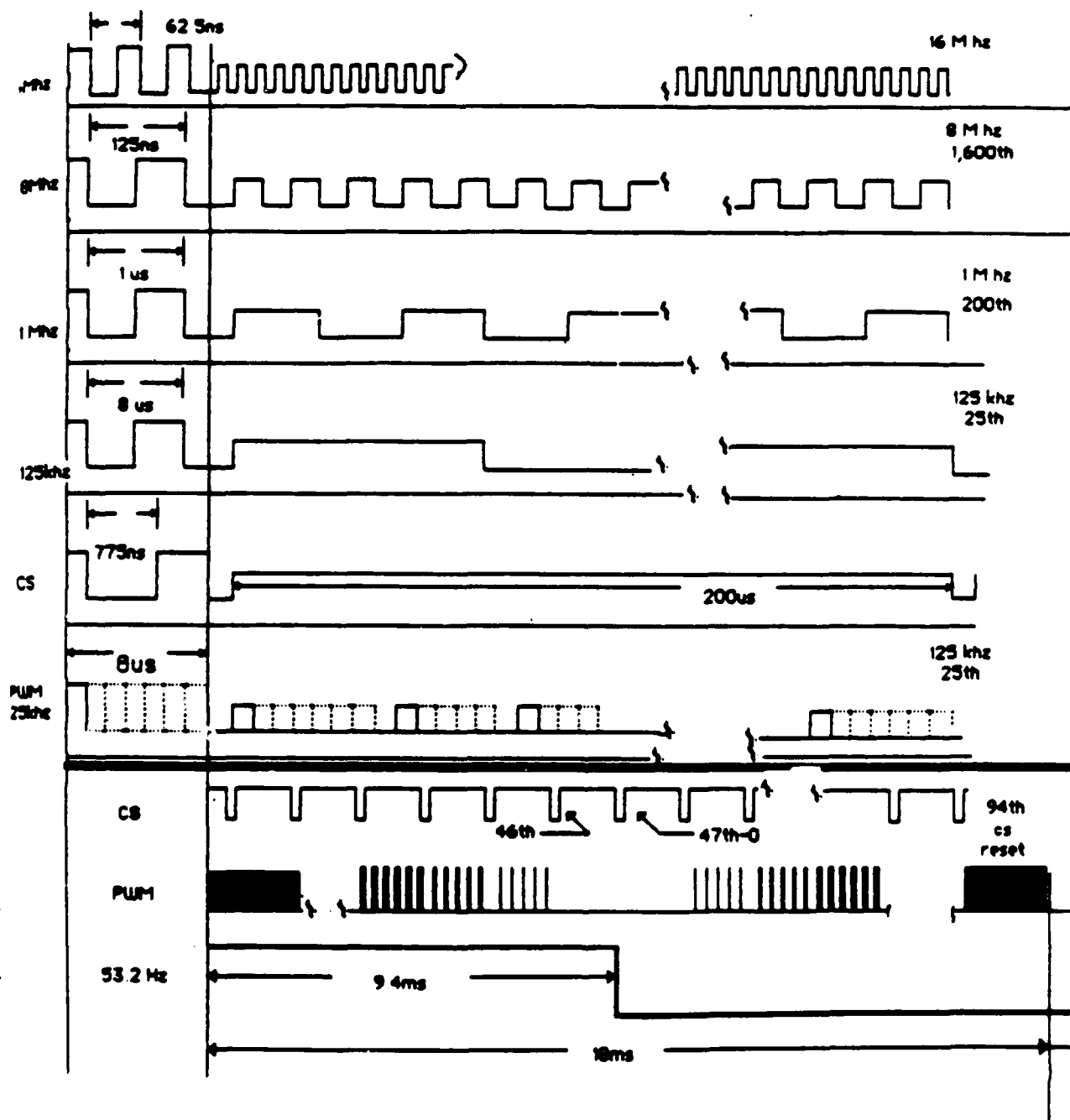


FIGURE 6. Timing Diagram

design is common to both the analog and the digital approach - no significant difference.

3.0 DESCRIPTION OF SOFTWARE

Algorithms were developed to implement open loop control of the cooler motor. A fixed input line voltage (24 Vdc) was assumed. A flowchart of this software is shown in Figure 7, with code, as developed using PL/M, listed in Appendix I.

Lookup Table (Table 2)

For the demonstration, the function of the computer was to update the DPWM chip with an 8 bit digital input. Basically, by means of the lookup table it is generating a digital sine wave command signal. The theory behind the development of the lookup table is very basic.

Analog Sine Wave: $V(t) = V_m \times \sin(2\pi f t) \times k \times d$

V_m = peak motor voltage (14 volts)

f = motor frequency (54 Hz)

$v(t)$ = instantaneous motor voltage

k = sample point

d = sampling interval

The voltage $v(t)$ is the sample value of the required cooler motor voltage at time $k \times d$. The sampling interval for demonstration was approximately 200 microseconds. Therefore the sine wave is approximated by steps that change every 200 microseconds, or 94 steps.

To generate the table therefore only the first half of the sine wave need be sampled or approximately 9.25 milliseconds for a 54 Hz sine wave. The values for a 14 volt peak or 9.89 Vrms sine wave are listed in column 2 of Table 2. The relationship between these voltage levels and the Pulse Width Modulator or Duty Cycle signal is as follows:

$v(t) = (T_{on}/T) \times V_{in} = PWM \times V_{in}$

PWM = pulse width modulation level (range 0 - 1.0)

T = switching period

T_{on} = on time of power switches

V_{in} = line voltage

Since we were operating open loop, the line voltage (V_{in}) was set to be constant at 24 Vdc. With this level established, the corresponding PWM levels are shown in column 3 of Table 2. These levels are then converted to the nearest hexadecimal number as listed in column 4 of Table 2.

SOFTWARE FLOWCHART

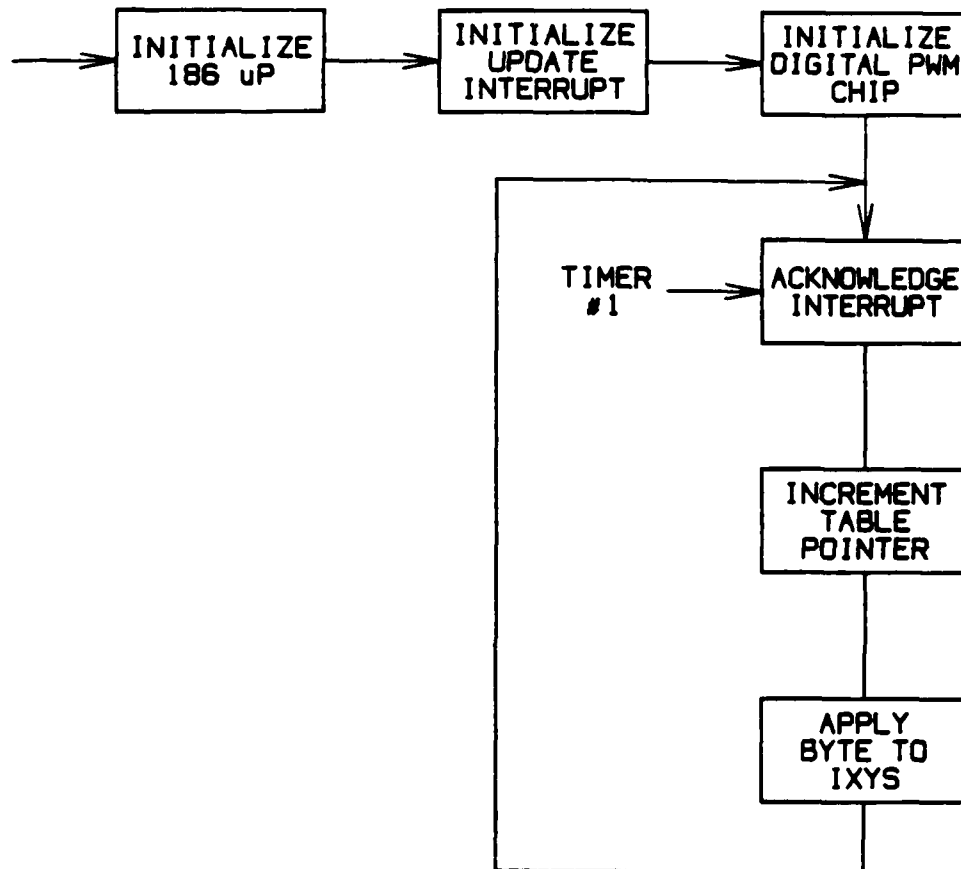


FIGURE 7

POINT	VOLTS (Vt)	DUTY CYCLE (PWM)	HEX NO.
0	0	0	000h
1	.9346235	3.894265E-02	00Ah
2	1.865077	7.771154E-02	014h
3	2.787209	.1161337	01Eh
4	3.696905	.1540377	028h
5	4.590107	.1912545	031h
6	5.462829	.2276179	03Bh
7	6.311178	.2629657	044h
8	7.131367	.2971403	04Dh
9	7.919739	.3299891	055h
10	8.672775	.3613657	05Eh
11	9.387114	.3911298	065h
12	10.05957	.4191488	06Ch
13	10.68715	.4452979	073h
14	11.26704	.46946	079h
15	11.79666	.4915276	07Fh
16	12.27365	.5114021	084h
17	12.69588	.5289948	088h
18	13.06146	.5442274	08Ch
19	13.36876	.5570318	08Fh
20	13.61642	.5673508	092h
21	13.80333	.5751385	093h
22	13.92864	.58036	094h
23	13.99181	.5829922	095h
24	13.99256	.5830233	094h
25	13.93087	.5804531	093h
26	13.80703	.575293	092h
27	13.62159	.5675661	08Fh
28	13.37537	.557307	08Ch
29	13.06947	.5445611	088h
30	12.70526	.5293858	084h
31	12.28436	.5118484	07Fh
32	11.80866	.4920273	079h
33	11.28026	.4700109	073h
34	10.70154	.4458975	06Ch
35	10.07507	.4197945	065h
36	9.403646	.3918186	05Eh
37	8.690269	.3620946	055h
38	7.938118	.3307549	04Dh
39	7.150549	.2979396	044h
40	6.331073	.2637947	03Bh
41	5.483353	.228473	031h
42	4.611168	.192132	028h
43	3.718409	.1549337	01Eh
44	2.80906	.1170441	014h
45	1.887177	7.863237E-02	00Ah
46	.9568709	3.986962E-02	000h

Table 2. Look Up Table

Note that on page 7 of the code listed in Appendix I, the lookup table is listed in accordance with Table 2. We emphasize again that this table is correct for 24 Vdc line voltage only and that in a final design will have to be scaled to accommodate a changing input line.

In addition to the lookup table, for the demonstration to be successfully implemented, algorithms were developed for the following functions:

1. Initialize Processor
2. Initialize Counter #1 - this counter controls the lookup table pointer - counts pulses at a 125 kHz clock and increments every 25 counts
3. Initialize PWM chip (IXDP610) i.e., deadtime set, lock bit off, 8 bit resolution.

4.0 SURVEY OF MICROCONTROLLERS

The heart of the microcontroller/digital approach to the motor driver electronics is the microcontroller. From the conception of the microprocessor in 1974 there have been many advancements in the VLSI technology of microprocessors. Improvements have steadily been made in performance such as processor speed, bus size and power consumption. However, probably most startling of all was the advancements in the ability of the IC manufacturers to pack more functions into a given area of substrate. It was the improvements made in the small size of the substrate that has made the evolution of the microcontroller possible.

A microcontroller differs from a microprocessor in that it is a self contained computer. Unlike a microprocessor it needs no support chips, and in fact, packs more function in the form of A/D converter, UART serial communication, interrupt controls and I/O. For these reasons Magnavox believes that a microcontroller is a natural progression for the next generation cooler electronics.

The major manufacturers of microcontrollers are: Intel, Motorola, National and Texas Instruments. Table 3, is a matrix that compares the existing microcontrollers and provide some of the primary features that relate to the cooler application. It should be pointed out that these features are as they exist at the time of this writing and because of the rapidly changing technology they would have to be reexamined again at the time of a design effort.

	INTEL	MOTOROLA	NATIONAL	TEXAS INSTRUMENTS
Family	MCS-8096	M68HC11	HPC16084	7000 Series
A/O	10 Bit	4 Channel, 8 Bit	Not Yet	No
Bus	16 Bit	16 Bit	16 Bit	8 Bit
PWM Output	Yes	No	Yes	No
Watchdog Timer	Yes	Yes	Yes	No
Timer	4	5	8	3
No. I/O Ports	5	5	5	-
Interrupt Sources	20	16	8	-
Power Consumption	100 mW	3 Modes Run - 50 mW Wait - 3 mW Stop - 1 mW	100 mW	100 mW
RAM	256 K	256 K	256 K	256 K
ROM	8 K	8 K	8 K	4 K
Price (Commercial)	\$16	\$16	Not Available	Not Available
MIL	Avail. In MIL	Will Be Avail.	Will Be Avail.	No
Software Development Status	Available	Available HDS-300	Available (Mole)	Available

Table 3. MICROCONTROLLER SURVEY

5.0 SELECTION OF SWITCHING FREQUENCY

5.1 VOLUME CONSTRAINT

The reason for the popularity of a PWM SWITCHMODE power driver is that this approach is very efficient. The theory behind this is that in the ideal, there are only two states for the switching devices - "on" and "off". In the "on" state we assume no voltage drop across the device and in the "off" state, no current flow. Therefore, in the ideal case no power is dissipated in the power amplifier. However, looking at the real device, there is a finite voltage drop when the device is "on" and a finite current when the device is "off". Furthermore there are switching losses when the devices change state. These losses increase with frequency.

This implies that from an efficiency point of view, it is best to keep the power amplifier switching frequency as low as possible. However, for the cooler electronics an overriding consideration is volume. For that reason, and as discussed earlier, the switching frequency must be above 100 kHz. Specifically, the size of the EMI components are frequency dependent and at frequencies much below 100 kHz, they do not fit.

5.2 DIGITAL HARDWARE CONSTRAINT

With the approach described we have hardware that can potentially switch up to 200 kHz. However, due to the requirements of the counters we have a practical constraint to choose frequencies that are divided by two, submultiples of the 16 MHz clock. This breaks down to frequency choices of 125 kHz, 62.5 kHz and 31.25 kHz. A frequency of 125 kHz was selected in accordance with the 100 kHz limitation described above.

5.3 THE SELECTION

At the present time 125 kHz is the best estimate of the optimum switching frequency. In future design efforts we would not expect any new inputs that would cause significant deviation from this frequency choice. However, the number of switching levels that go into the sine wave may vary since this number may influence the vibration profile of the cooler.

6.0 SUMMARY

The work carried out in this study was performed by Magnavox Government & Industrial Electronics Company, Electro-Optical Systems, in accordance with the requirements of contract DAAB07-87-C-F018. The purpose of the program was to conduct studies of microcontroller/digital VLSI technologies for possible future use in Magnavox cryogenic coolers. In pursuit of this

objective, the following specific tasks were required of the study:

Task No. 1: Study of Single Chip Microcomputers

Task No. 2: Determine Optimum Pulse Width Modulation Frequency

Task No. 3: Develop Algorithms

Although the scope of this study was initially conceived to be a white paper report/study only, Magnavox has taken the initiative to implement the basic concepts with a hardware demonstration. That is, a microcontroller was simulated using an Intel 80186 microprocessor and configured to drive an operating cooler. This clearly substantiates the microcontroller/digital approach and is planned as a demonstration for cognizant CNVEO personnel.

7.0 RECOMMENDATIONS FOR FURTHER STUDY

Based on the positive findings of this research effort it is recommended that additional funding be provided to implement a high switching frequency microprocessor/digital design program to further refine the cooler electronics. Specific tasks recommended for investigation include:

1. Investigation of nonsinusoidal motor voltage drive waveforms. This would be implemented by appropriate changes in the software. The potential benefits from this investigation would be lower vibration levels and increased operating efficiency of the cooler.
2. Investigate "SMART" cooler electronics. Self calibration, self test and diagnostics are a few of the possibilities to be explored.
3. Develop optimum stepped sinusoidal waveform. This will require careful analysis before establishing a firm number for the design.
4. Explore use of feedback signals representing the working gas temperature or pressure. Determine the effects of these signals on cooler performance, i.e., stability of temperature set point, efficiency and vibration.

APPENDIX I

DEMONSTRATION SOFTWARE

IBM PC-DOS PL/M-86 V2.3 COMPILATION OF MODULE INT186
 OBJECT MODULE PLACED IN INT186.OBJ
 COMPILER INVOKED BY: PLM86 INT186.PLM DEBUG XREF CODE MOD186

```

1      INT186:

      DD:
2      1      DECLARE FLAG BYTE EXTERNAL;

      /*****
      STARTING OF INITIALIZTION OF APX186, AND 2681 QUART'S
      UMCS is initialized in asm program STRTSEG at OFFF0h
      *****/

3      1      Declare
      LMCS_REG  literally 'OFFA2h', /* Chip select register locations*/
      PACS_REG  literally 'OFFA4h',
      NPXS_REG  literally 'OFFA8h',
      MLOC_REG  literally 'OFFA6h';

4      1      Declare /* Timer control registers locations*/
      t2mode_reg  literally 'OFF66h', /* Timer 2 mode control word */
      t2maxa_reg  literally 'OFF62h', /* Timer 2 max counter A */
      t2count_reg  literally 'OFF60h', /* Timer 2 counter register */
      t1mode_reg  literally 'OFF5Eh',
      t1maxB_reg  literally 'OFF5Ch',
      t1maxA_reg  literally 'OFF5Ah',
      t1count_reg  literally 'OFF58h',
      t0mode_reg  literally 'OFF56h',
      t0maxB_reg  literally 'OFF54h',
      t0maxA_reg  literally 'OFF52h',
      t0count_reg  literally 'OFF50h',
      timer_int  literally 'OFF32h';

5      1      Declare /* DMA control registers locations */
      dma0_cntrl_reg  literally 'OFFC0h', /*DMA channel 0 control word */
      dma0_count_reg  literally 'OFFC8h', /*DMA channel 0 transfer counter*/
      dma0_dest_H_reg  literally 'OFFC6h', /*DMA channel 0 destination high ptr*/
      dma0_dest_L_reg  literally 'OFFC4h', /*DMA channel 0 destination low ptr */
      dma0_src_H_reg  literally 'OFFC2h', /*DMA channel 0 source high pointer */
      dma0_src_L_reg  literally 'OFFC0h', /*DMA channel 0 source low pointer */

      dma1_cntrl_reg  literally 'OFFD0h',
      dma1_count_reg  literally 'OFFD8h',
      dma1_dest_H_reg  literally 'OFFD6h',
      dma1_dest_L_reg  literally 'OFFD4h',
      dma1_src_H_reg  literally 'OFFD2h',
      dma1_src_L_reg  literally 'OFFD0h';

6      1      Declare
      t0mode  literally '0C003h',
      t1mode  literally '0E005h',
      t2mode  literally '0C001h', /* en/inhcount true */
      c1mode  literally '014F7h';

      /----- set up for PACS chip select ----- */

```

```

7 1  Declare
    MCS_SIZE  literally '00BFh', /* pacs size is 00,programed only to set */
                /* peripheral map to I/O space & A1-A2 not */
                /* latched.*/
    PACS_LOC  literally '013Dh'; /* peripheral chip select base addr = 1000h*/
                /* one wait states for pacs0-3; 3-wait for 4-6*/
                /* external ready ignored for all pacs's      */

/*----- end of equates -----*/

8 1      init186: PROCEDURE PUBLIC;

9 2      Outword(MPCS_REG) =MCS_SIZE; /*init midrange block size */
10 2     Outword(PACS_REG) =PACS_LOC; /*init peripheral chip select */

/* ----- set up time 2 for RAM refresh -----*/

11 2     Outword(t2mode_REG) =t2mode; /*timer 2 set for RAM refresh */
12 2     Outword(t2maxA_REG) =32D;    /*timer 2 set for 16 u/s rat */
13 2     Outword(t2count_REG) =00h;

/* ----- set up time 1 for counting 200us -----*/
/* Counter-1 is set up to count 25 125khz Pulses then start a
   interrupt for a count of 200us */

14 2     Outword(timer_int) = 0000;
15 2     Outword(t1mode_REG) =t1mode;
16 2     Outword(t1maxA_REG) =019h;    /* 25 counts */
17 2     Outword(t1maxB_REG) =00000h;
18 2     Outword(t1count_REG) =00h;

/* -----set up DMA CHANNEL 1 for RAM refresh -----*/
/* DMA is programmed for a memory-to-I/O transfer, with writes
   to nonexistent I/O space. The DMA is programmed to continuously
   run through the entire megabyte of memory at word transfers */

19 2     Outword(dma1_dest_H_reg) =0FFh;
20 2     Outword(dma1_dest_L_reg) =0FFh;
21 2     Outword(dma1_src_H_reg) =00h;
22 2     Outword(dma1_src_L_reg) =00h;
23 2     Outword(dma1_cntrl_reg) =c1mode; /* dest,mem,no inc/dec */
                /* source,I/O,inc joff-tc-int */
                /* SYN-unused; on-P-TDRQ-CH6-ST-word */

24 2     END; /* INIT186 PROC */
/*-----
   *          SETUP PROCEDURE OF THE 2681 QUART UTILITY
   *-----*/

25 1     INIT_2618:
        PROCEDURE PUBLIC;

```

```

26 2  DECLARE TEST BYTE ;
27 2  DECLARE
      INT_MASK    literally '0FF20h', /* 90186 interrupt */
      INTO_CONTROL literally '0FF30h';

28 2  DECLARE
      MODE_REG_0  literally '1000h', /* (MR1A,MR2A) */
      CLOCK_SEL_0 literally '1002h', /* (CSRA) write */
      STATUS_0    literally '1002h', /* (SRA) read */
      COMMAND_0   literally '1004h', /* (CRA) write */
      TX_DATA_0   literally '1006h', /* (THRA) write */
      RX_DATA_0   literally '1006h', /* (RHRA) read */
      AUX_CONTROL_0 literally '1008h', /* (ACR) write */
      INTRERRUPT_0 literally '100Ah', /* (IMR) write */
      COUNT_UPPER_0 literally '100Ch', /* (CTUR) write */
      COUNT_LOWER_0 literally '100Eh', /* (CTLR) write */
      MODE_REG_1  literally '1010h', /* (MR1B,MR2B) */
      CLOCK_SEL_1 literally '1012h', /* (CSRB) write */
      STATUS_1    literally '1012h', /* (SRB) read */
      COMMAND_1   literally '1014h', /* (CRB) write */
      TX_DATA_1   literally '1016h', /* (THRB) write */
      RX_DATA_1   literally '1016h', /* (RHRB) read */
      COUNT_CONTROL literally '101Ah', /* (OPCR) write */
      OUT_RESET   literally '101Ch'; /* reset write */

/* Current setup is channel a: even parity: 8 data bits: normal op:
   RTS on CTS on : 2 stop bit : baud rate is 9,600 */

29 2  Declare
      SET_MASK literally '00h', /* enable 186 int-0 and TMR */
      MODE1    literally '83h', /* even parity 8 data bits */
      MODE2    literally '3Fh', /* normal TxRTS CTS en 2 STOP */
      BAUD     literally '08Bh', /* 9600 baud rate */
      AUX_CR   literally '0h', /* disable the aux_control */
      IMR      literally '2h', /* enable RxRDY int reg */
      OUTPUT_PORT literally '04h', /* Output Port Configuration Reg*/
      UPPER_COUNT literally '00h',
      LOWER_COUNT literally '04h'; /* divided 8.0 Mhz by 8 = 125 kHz*/

30 2  Declare TRUE literally '0FFh', FALSE literally '0';

31 2  DO;
32 3  ENABLE;
33 3  OUTPUT(COMMAND_0)= 1010h;
34 3  OUTPUT(MODE_REG_0)=MODE1;
35 3  OUTPUT(MODE_REG_0)=MODE2;
36 3  OUTPUT(CLOCK_SEL_0)=BAUD;
37 3  OUTPUT(INTRERRUPT_0)=IMR; /* interrupt mask register */
38 3  OUTPUT(COMMAND_0)= 15h; /* ENABLE Rx and TX */
39 3  OUTPUT(OUT_RESET)= 01h; /* RESET OUT */
40 3  OUTPUT(INT_MASK)= SET_MASK;
41 3  OUTPUT(INTO_CONTROL)=10h; /* LTM,PRO: Highest.C: direct,MSK: none */
42 3  OUTPUT(COMMAND_1)= 1010h;
43 3  OUTPUT(MODE_REG_1)=MODE1;
44 3  OUTPUT(MODE_REG_1)=MODE2;
45 3  OUTPUT(CLOCK_SEL_1)=BAUD;

```

```
46 3      OUTPUT(COMMAND_1)= 15H; /* ENABLE Rx and TX */
47 3      OUTPUT(COUT_CONTROL)= 04H;
48 3      FLAG = 0; /* set char flag to no character */

/* ----- SET UP FOR 16Mhz CONTROL REG ----- */

49 3      OUTPUT(01200h) = 080h;

/* ----- SET UP DUART TIMER ----- */

/* DUART Timer set to output 125khz this is acheved by takeing the 8Mhz
system clock and deviding it by 8 for a 1 Mhz clock into the DUART timer
their it is devided by 8 for a output of 125khz */

50 3      OUTPUT(AUX_CONTROL_0)=040h; /* MODE SET FOR EXTERNAL (IP2) */
51 3      OUTPUT(COUNT_UPPER_0)= UPPER_COUNT; /* UPPER_COUNT; */
52 3      OUTPUT(COUNT_LOWER_0)= LOWER_COUNT; /* LOWER_COUNT; * 2 = 8 */
/* ----- */

53 3      END; /* SETUP */
54 2      END INIT_2618; /* init 2618 duart */

55 1      END INT186;
```

```

156 2      END GETCH;

/*-----*/
/*          PWM_SET_UP          */
/*-----*/
/* this function is used to program the IXYS IXDP610 Digital Pulse Width
   Modulator to the fowing operating control byte */
157 1      SETUP_PWM: PROCEDURE PUBLIC;

158 2      DECLARE SET_PORT ADDRESS;
159 2      DECLARE CONTROL_LATCH literally '1302h';
160 2      DECLARE (CONTROL_BYTE, PWM_BYTE) WORD;      /* BYTE;*/

/*          PWM_CONTROL_BYTE
   7-6-5-4-3-2-1-0
           0-0-0   Dead time (0)
           0----- Not Used
           0----- Lock Bit (off)
           0----- Divide Bit (no division)
           1----- Resolution Bit (8-bit)
           1----- Stop Bit (output enabled) */

161 2      SET_PORT = 0;
162 2      CONTROL_BYTE = 0c0h;      /* set the control byte */

163 2      SET_PORT = CONTROL_LATCH; /* Set address of control port */
164 2      OUTPUT (SET_PORT) = CONTROL_BYTE; /* output control byte */
165 2      COUNT1 = 0; /* clear counter-0 to zero */
166 2      COUNT2 = 1; /* set counter-1 to 1 witch brings the cs in line */

167 2      END SETUP_PWM;

/*-----*/
/*          PWM_UP DATE          */
/*-----*/
/* This procedure is interrupt driven by Timer 1 wich is used to count a */
/* 125khz 8u/s signal it is this timing that sets when the interrupt */
/* is to occur */
/*-----*/

168 1      UPDATE_PWM_INT:
        PROCEDURE INTERRUPT 18 PUBLIC;

169 2      DECLARE PW_LATCH    literally '1300h'; /* address of pwm latch */
170 2      DECLARE EOI_REGISTER literally '0FF22h'; /* End Of Interrupt resets 15*/

171 2      DECLARE PWM_DATA WORD;
172 2      DECLARE HIGH_WORD DATA (0200h); /* set bit 9 to one */
173 2      DECLARE LOW_WORD DATA (0); /* set bit 9 to zero */
174 2      DECLARE LEVEL WORD;

175 2      DECLARE PWM_TABLE (*) WORD DATA
        (000h,00Ah,014h,01Eh,028h,031h,03Bh,044h,04Dh,055h,
         05Eh,065h,06Ch,073h,079h,07Fh,084h,088h,08Ch,08Fh,
         092h,093h,094h,095h,09Ah,093h,092h,08Fh,08Ch,088h,
         084h,07Fh,079h,073h,06Ch,065h,05Eh,055h,04Dh,044h,
         03Bh,031h,028h,01Eh,014h,00Ah,000h);

```

```
176 2      LEVEL = 00;
177 2      IF COUNT1 = 47 THEN          /* is counter-1 at senter of 53 hz */
178 2          COUNT1 = 0;              /* Reset counter at end of table */
179 2      IF COUNT2 = 47 THEN          /* is counter-2 at senter of 53 hz */
180 2          LEVEL = HIGH;            /* then set bit 9 high */
181 2      ELSE                          /* else counter-2 is set to low side*/
182 2          LEVEL = LOW;              /* of 54hz set bit 9 to low */
183 2      IF COUNT2 = 94 THEN
184 2          COUNT2 = 0;              /* reset counter-2 at end of 53Hz cycle */
185 2      PWM_DATA = LEVEL OR PWM_TABLE (COUNT1); /* add level and data */
186 2          OUTWORD (PW_LATCH) = PWM_DATA; /* output data and level */
187 2      COUNT1 = COUNT1 + 1;          /* update counter 1 */
188 2      COUNT2 = COUNT2 + 1;          /* update counter 2 */
189 2
190 2      OUTWORD (EOI_REGISTER) = 8000h; /* End Of Interrupt resets */
191 2
192 2      END UPDATE_PWM_INT;
193 2
194 2      /*****
195 2      STOP: PROCEDURE PUBLIC;
196 2          HALT;
197 2      END STOP;
198 2      /*****
199 2      MAIN:
200 2          DO;
201 2          HALT;
202 2          END;
203 2      END MAIN_ROUTINE;
```